Online Research @ Cardiff

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository: https://orca.cardiff.ac.uk/id/eprint/119120/

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Doe, Robert ORCID: https://orcid.org/0000-0002-8246-3003 2018. Facilitating integration of computational design processes in the design and production of prefabricated homes. Architectural Science Review 61 (4) , pp. 246-254. 10.1080/00038628.2018.1466686 file

Publishers page: http://dx.doi.org/10.1080/00038628.2018.1466686 <http://dx.doi.org/10.1080/00038628.2018.1466686>

Please note:
Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See http://orca.cf.ac.uk/policies.html for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.

information services
gwasanaethau gwybodaeth

Facilitating Integration of Computational Design Processes in the Design and Production of Prefabricated Homes

This research utilises computational design concepts to address dysfunction in the architectural, engineering and construction (AEC) sector. The goal is to identify solutions by focussing on the integration of computational design concepts into computer-aided design (CAD) processes and into compatible prefabrication strategies.

Precedents are examined and project work is used to set up the modelling framework of a three-storey, prefabricated residential building, allowing reflection and speculation on solutions to the problems identified. The computational design concept of the 'design domain' is employed to assist with set up of the modelling framework, while computational 'design rules' are implemented during the developed design stage, including the 'declarative format' which improves accessibility using visual programming graphs, and the 'modular format' which reveals combined effects between programming and prefabrication strategies.

The intensity of integration between these computational design, AEC CAD and prefabrication processes is assessed, allowing solutions to be developed which could improve conventional AEC CAD practice and traditional AEC prefabrication processes.

Keywords: integration; computational design; AEC CAD; prefabrication

INTRODUCTION

This study identifies problems with existing AEC CAD and prefabrication processes which are fragmented and ineffective. The hypothesis affirms that computational design concepts are needed to foster the integration of AEC CAD and prefabrication processes. Thus, the research questions addressed:

- How computational design concepts can instil understanding, in concert with current AEC CAD practices, and how they might become integrated with everyday architectural design workflows.
- How prefabrication can benefit from better understanding and incorporation of these computational design concepts.

This research reflects on project work undertaken to gain a better understanding of computational design processes incorporated into AEC CAD and compatible prefabrication strategies. It is contended that, to achieve the goal of integrated computational design, AEC CAD and prefabrication processes, fundamental and systemic changes are required. This prompts a proposal for evaluating integration intensity to facilitate the formulation of solutions.

METHODOLOGY

The strategy employed in this research has been the study of precedents followed by investigation through project work, testing the main hypothesis that integrated computational design concepts are needed to ensure effective implementation of design tasks which, acting in concert with compatible prefabrication strategies, can also improve the design and production of prefabricated homes. In summary, the methodology followed in this paper is engaged with:

- Revealing through a literature review the complementarity of computational design concepts combined with prefabricated home design and production strategies.
- Substantiating through project work that:
    i. When setting up the framework of a computationally designed model, integration is improved through better understanding of the ‘design

domain', and by implementation of more flexible and intelligible 'design rules'.

ii. There are synergies between the programming concept and the prefabrication concept of the *modular format*, so that working together, design and production workflows are enhanced.

- Recording the findings of this research process through 'reflection-in-action', the act of noticing while practicing to capture the subtleties of concepts, methods and tools used (Schön 2008, 85).
- Advocating *Mapping Integration,* introduced by the author as a means of evaluating the integration intensity of computational design, AEC CAD and prefabrication processes implemented during everyday architectural design tasks.

The objective of the externally funded research stream was to compare conventional with prefabricated methods of designing and producing homes. Hence, project work carried out as an adjunct to the research stream facilitated this objective and tested the main hypothesis.

Literature Review

Fragmentation of AEC Systems

The AEC sector has failed to develop design and production tools that implement objectives in an effective way. As Ryan Smith notes in *Prefab Architecture, a Guide to Modular Design and Construction,* the sector has yet '…to deliver a product that meets the requirements of design, on budget, on time, without falling down or leaking' (Smith 2010, viii). Integrated computational design and production tools offer solutions to these problems and can bring the AEC sector forwards into the digital age

with its expectations of collaboration and seamlessness, as noted by Mario Carpo in *The Alphabet and the Algorithm* (2011, 79).

Interoperability, the effective exchange of data and communication, is a key component of integration which needs improving in the AEC sector, despite the efforts of Building Information Modelling (BIM), a concept initiated by Charles Eastman et al, at Carnegie Mellon University in 1974 (Eastman et al. 1974). Currently, BIM's focus is to improve AEC management processes for the collection of all project information (Aconex 2017). But this research asserts that it is improved technological processes which will have a greater impact on interoperability, as integration of computational design concepts into AEC CAD processes will facilitate a seamless workflow from conceptual design to production.

Sustaining the emphasis in this research on technological solutions, the Discussion section proposes a means to measure the integration intensity of everyday architectural design tasks incorporating computational design, AEC CAD and prefabrication processes.

Computational Design

Computational design methods may be merged with or separate from current AEC CAD practices, but they have the distinguishing tendency to cultivate 'intentionality', or '…the mapping of an idea through to an intended outcome', as noted by Mark Burry (2011, 25). Indeed, incoherently defined AEC CAD practices have been attributed to architects' lack of engagement with its development since the early 1960's, leaving the technological capabilities of CAD systems to be resolved by programmers, to the extent that '…CAD software developers are meta designers…'

(Terzidis 2006, 54). Consequently, in many architectural practices, CAD is mainly used to implement 'computerisation', a pejorative term, which Terzidis explains as follows:

> Generally, it involves the digitisation of entities or processes that are preconceived, predetermined, and well defined (Terzidis 2006, 57).

By contrast, as observed by Yehuda Kalay (2004, 195), computational design facilitates the embedding of rules, constraints and goals within modelled objects, thus utilising the power of the computer and offering itself as a more intelligent aid in design and production workflows than current AEC CAD.

Prefabrication

Prefabrication also provides the AEC sector with an opportunity to improve the design and production process. As perceived by Smith et al (2010, xii), it is an approach which, compared to conventional building practice, will '…allow for greater efficiency and precision'. Though there are many definitions of prefabrication Nicholas Habraken's broad description from *Supports: an alternative to mass housing* (1999) is instructive because it encompasses the whole spectrum of definitions, and is thus accommodating rather than confining:

> In itself prefabrication means no more than the manufacture of housing components in one place and their assembly in another (Habraken 1999, p.67).

Prefabrication has a long history in the AEC sector, but architects' engagement with it has been irregular and often unsuccessful. It has been argued by Colin Davies in *The Prefabricated Home*, that this has been due to a lack of engagement with industry, while Gropius' and Wachsmann's *Packaged House* exemplified this failure (Davies 2005, 25). Seemingly, architects have preferred to engage with the AEC sector through the implementation of dimensionally coordinated systems. For example, Rudolph

Schindler's preoccupation with modular coordination in the 1920s and 30s was an early attempt to integrate design and construction (Park 2005).

More recently, Kieran and Timberlake in *Refabricating Architecture* have called for the '…development of integrated component assemblies – modules, chunks, grand blocks', where the contractor becomes an assembler of components on site (Kieran and Timberlake 2004, 41). It is this approach to prefabrication using 'modular component assemblies', implemented in conjunction with compatible computational design concepts, which is examined and evaluated in this research.

Computational Design Concepts

Design domain. Project work examines set up of the framework of the computationally designed model based on the concept of the *design domain*. This is described by Jane Burry as a framework which creates better understanding of the structure of the artefact being designed because it is a topological space where associations are abstracted from geometrical form (Burry 2007, 615). Associations in topological space are often exemplified by visualisation of the smooth transformation from torus to mug. Though the associations of this abstracted framework create distance between the architect and design intent, Burry asserts that it allows the deferral and later review of design decisions, and even provides an expanded cognitive design space for exploration by students and architects (Burry 2007, 622).

Design Rules. In the quest for more flexible and intelligible computational *design rules* it has been argued by William J Mitchell in *A New Agenda for Architectural Design* (1989, 8), that they should be expressed in a 'declarative' and 'modular' format, concepts common to the field of computer programming. This research examines this quest to improve the format in which *design rules* are expressed, while the rules

themselves are described in more detail as follows:

Design rules in *declarative format* specify how particular relationships between objects are processed, but they do not describe how the outcome is achieved. In contrast, as Kalay explains (2004, 50-53), 'imperative format' *design rules* '…are languages where each action of the computer is spelled out and in the correct sequence'. The difference between these formats is further illustrated by the user interface which represents them: *declarative formats* are represented by *visual programming graphs* e.g. Generative Components by Bentley, Dynamo by Autodesk and Grasshopper by David Rutten, while *imperative formats* are represented by textual programming languages e.g. C# and Python.

Design rules in *modular format* denote discrete input and output parameters, as noted by Daniel Davis et al (2011, 58) with modules which comprise 'self-contained chunks of code', facilitating sharing, re-use, and debugging. In describing component-based software engineering best practice, Wilhelm Hasselbring (2002, 2) also defines the features of the programming concept of the *modular format* in more detail as follows:

- *Reusability*. Component reuse in multiple applications to lower costs.
- *High cohesion.* Cohesion of a component is the extent to which contained elements are inter-related. For the component to be self-contained or distinctive high cohesion is desired.
- *Interface coupling.* Coupling is the extent to which the component is coupled with other components. The component's interface should be loose, thus low coupling is desired.
- *Trade-offs.* Optimal performance involves trade-offs between many small components, or a few larger components.

Connections between the programming concept of the *modular format* and the prefabrication concept of the *modular format* - i.e. *modular component assemblies*, as noted earlier - are also examined in this study.

Integrated Systems

This review has highlighted the fragmentation and inefficiencies which exist in AEC CAD and prefabrication processes, and consequently the necessity for a better understanding of successfully integrated systems to develop solutions to these problems. For example, when elucidating lessons from Ludwig von Bertalanffy's (2008) *General System Theory*, Barry Russell in *Building Systems, Industrialisation and Architecture,* suggested that the 'interacting forces' within an open system should be acknowledged:

> … intuitive or explicit, there has to be a recognition by architects of the interacting forces in an open system. Using this open system in a responsive and responsible way can offer new and original results as well as drawing sensitively on tradition (Russell 1981, 705).

Furthermore, in *The Sciences of the Artificial*, Herbert Simon maintained that a hierarchically structured complex system, which discloses dynamic interactions between its parts in relation to the whole, is a structural organisation which facilitates understanding (Simon 1996, 207). Simon correlated this 'process description' of complex systems with an organism's 'capacity for acting purposefully on its environment' (Simon 1996, 215). These insights suggest that an open, dynamically interactive design and production system would lead to a more successfully integrated, effective and adaptable outcome for the AEC sector.

Accordingly, a methodology termed *Mapping Integration*, is proposed to improve integration of computational design, AEC CAD and prefabrication processes. It asserts that the features of this integrated system should include:

- Openness: interoperability, collaboration
- Dynamic interaction: interactivity and operability

These features and their attributes guide reflection of the Findings, while the methodology for *Mapping Integration* is described in more detail in the Discussion section.

Summary

The literature review has examined the state of fragmentation and ineffectiveness of AEC CAD and prefabrication processes in general, and has explored precedents which offer solutions to these problems. Computational design concepts which could improve these processes have been described, including the *design domain,* and *design rules* which utilise the *declarative* and *modular formats*. These computational design concepts are integrated with AEC CAD processes and compatible prefabrication strategies in an approach which underpins the project work examined in the Findings which follow.

FINDINGS

The primary problem examined in this research is the challenge of integrating computational design concepts into AEC CAD and prefabrication workflows to improve outcomes. Initially this involved set up of the framework of the computational design model applying the concept of the *design domain*, and was followed during the design development stage with an examination of the *declarative* and *modular formats*' ability to improve understanding and intelligibility. CAD tools used to implement project work and facilitate comparison of findings included: 'AEC CAD' (Autodesk Revit); a 'visual programming graph' (Autodesk Dynamo); and 'advanced CAD' (Digital Project/ CATIA), used primarily in the manufacturing sector.

The secondary problem examined in this research is the challenge of implementing compatible computational design, AEC CAD and prefabrication strategies. Different options for prefabrication were evaluated by the research team and a hybrid option - panellised walls and floors, plus 3D volumetric components - was determined to provide the least cost and highest design flexibility. Project work focussed on design development of the panellised walls and floors.

The intensity of integration of the computational design concepts was guided by reflection on the features outlined by *Mapping Integration* described earlier.

The Design Domain

With the building's envelope previously established following discussions between stakeholders and planners, project work focussed on set up of the framework of the *design domain*'s parts in relation to the whole. The framework of the whole was defined by grids and levels associated with 'components' built-in to the AEC CAD software, a tool categorised by Schodek et al (2005, 184) as a 'component based modeller' (e.g. Autodesk Revit, Bentley Microstation, Graphisoft Archicad). After adjustment of grids and levels, 'grouped' modules – residential units, bathroom pods, clip-on balconies - were repeatedly copied into grid locations to facilitate parametric propagation of changes (Figure 1).

Figure 1. Typical set-up AEC CAD: 'grouped' modules - unit, bathroom pods, clip-on balconies (Autodesk Revit).

The AEC CAD process involved an efficient and widespread architectural workflow relying on mouse-click placement of implicitly defined 'components' - predefined wall and floor assemblies. For example, predefinition determined that walls were joined perpendicularly based on their centreline, interior face or exterior face location, not in accordance with the 'component's' assembled 'parts' or construction layers. Hence, the

method for joining ‘components’ was implicitly defined by the software engineer’s assumptions for best practice, rather than explicitly defined by the user or architect.

Templates – explicit set up of the framework

An alternative computational design approach, using the same AEC CAD tool, set up the *design domain* of a ‘component family’ as a template which facilitated the creation of varied wall and floor panel types. With this approach, the origin for placement and joining of components could be explicitly defined, which facilitated automated procedures used later on. Thus, with this approach, a stronger interactive parametric relationship was defined between components (Figure 2).

Figure 2. Floor panel design domain - a template for floor panel types (Autodesk Revit).

Describing a process, not a product

Jane Burry (2007, 615) suggested that a better method for defining the *design domain* is a textual rather than a graphical ‘sketch’ of associations and relationships between elements, because it focusses attention on the process, rather than the product. This method was tested by defining the wall template’s relationships textually as follows:

(1) Instantiate the wall panel to any thickness, width or height.
(2) Allow width interfaces to be notched or un-notched.
(3) Provide an origin located to allow for automatic placement of the wall panel in the model’s framework.
(4) Enable the location of an opening of any height or width to be located in the panel.
(5) Ensure fixing brackets are automatically located at 600mm centres, set-in 300mm at the panel’s ends.

(6) Ensure that all elements' data can be automatically scheduled.

This method concentrated awareness on set up of the template and all likely interface conditions which could be predicted in advance (Figure 3).

Figure 3. Wall panel component design domain: reference planes, associated dimensions, and geometry (Autodesk Revit).

Setting up the *design domain* with this method necessitated a logical sequence of steps: firstly, reference lines and planes were sketched out; then, dimensions were associated to these entities; lastly, geometry was aligned, allowing parametric relationships to be established. Such rigour is characteristic of the extra cognitive effort required of computational design methods. Despite this cost, the benefit of carefully defining and dynamically engaging components instantiated from the wall and floor panel templates, was a reduction in the complexity of the whole model. This occurred because templates minimised the amount of drafting required by maximising the potential for re-use and sharing of the knowledge embedded in the templates themselves.

Generally, explicit and logical set up of the framework of templates facilitated visualisation and understanding of relationships between topological, geometrical and parametrically associated entities.

The Declarative and Modular Formats

The *declarative* and *modular formats*, encapsulated by the *visual programming graph*, tested Mitchell's notion (1989, 8) that these formats could improve the flexibility and intelligibility of computational design concepts. The *declarative format* facilitated explicit definition of grids and levels using nodes and wires to establish parametric relationships between graph and model, thus enhancing interactivity. The grid

intersections were then used to define coordinate locations, offering the following benefits (Figure 4):

- A record of decisions, and therefore the ability to review or defer them
- An ability to search for grid intersections, and therefore extract coordinates
- A high degree of precision in the placement of components, facilitating the planning of downstream production outcomes

Figure 4. Set-up of the model's design domain to find grid intersection coordinates (Autodesk Dynamo).

When the extracted coordinate values were projected to floor levels, the *visual programming graph* enabled automated placement and rotation of wall and floor panels, based on their origin points, although operability was impaired when errors occurred during placement as some of the graph's nodes malfunctioned (Figure 5).

Figure 5. Automatic placement and rotation of wall and floor panels based on coordinates projected to locations at each level (Autodesk Revit & Dynamo).

As implied above, the programming concept of the *modular format* was also represented by the features of the *visual programming graph*. For example, the nodes were self-contained modules which facilitated sharing and re-use as they were copied and amended to define a new object, or process relationship. These node modules represented the wall and floor panels which were referenced into the *visual programming graph* from the AEC CAD model templates described above, facilitating automatic placement of these panels into the CAD model.

Modular format programming and prefabrication

Links between *modular format* programming and *modular format* prefabrication concepts showed that the notion of the interface of the module differed markedly. As observed by Hasselbring (2002, 2), at the interface between programming modular

components low coupling is desirable, whereas with prefabricated modular components, whilst low coupling is likely interface conditions change significantly according to the context of the assembly. For example, floor to floor panel interfaces needed to maintain fire and acoustic performance, transfer loads continuously, make an allowance for movement, or even required full separation where inside met outside at balcony locations (Figure 6).

Figure 6. Floor & wall panel interfaces (background details by Kate Humphries, UQ).

Nevertheless, the *modular format*, which aimed to improve the flexibility and intelligibility of programming modules, correspondingly improved definition of *modular component assemblies* by promoting self-containment and distinctiveness, reusability and modifiability, and well-defined interfaces.

Trade-offs

Reusability and modifiability were partly dependent on the wall and floor panel templates encompassing all the modular components' interface possibilities, as defined by their fixed and variable parameter values. But, a trade-off between complexity and flexibility emerged where many templates could have different, fixed interfaces, or one template could have many, variable interfaces. Hasselbring (2002, 4-6) observed that, in software programming, trade-offs between numbers of components, and regulation of complexity through their reuse or instantiation as types, is supported by 'domain engineering' managed by a component library. Similarly, *modular component assemblies* could benefit from such careful management of trade-offs.

Hierarchical structure and the modular format

Using advanced CAD software, *design domains* were recreated, and the *modular format* of prefabrication was re-examined. The graphical user interface (GUI) of

advanced CAD differed from AEC CAD and *visual programming graphs* because parametric associations were explicitly formed and managed via hierarchical 'trees'. Furthermore, the cognitive burden was increased when setting up frameworks because components were defined explicitly as 'solid' entities containing volumes and physical properties. For example, the *design domain* was defined by 'sketches', parametrically associated to geometry, which were comprised of extrusions, pads and pockets related to each other as solids and voids. This meticulous process is known as 'feature-based' modelling in the manufacturing sector (Schodek et al. 2005, 202). Therefore, the wall panel component was made up of three sketches - an opening, notches, and the 'pad' or panel itself - features which could then be examined in the hierarchical tree where their parameters were defined and modifiable (Figure 7). Hence, with advanced CAD, intelligibility and speed are traded for rigour, as observed by Schodek et al (2005, 185).

Figure 7. Framework sketches, 'feature based' product assembly & hierarchical tree (Digital Project/CATIA).

Accordingly, wall panel and spaced bracket 'parts' were set up as *design domains,* or templates, for instantiation of all *modular component assemblies*. Hence, the set up of these discrete, self-contained, re-usable components linked *modular format* programming with *modular format* prefabrication concepts (Figure 8).

Figure 8. Grids for each CLT wall and floor panel component, Glulam beam, and automated wall bracket placement to unit type (Digital Project/CATIA).

Summary

The Findings examined the integration of computational design, AEC CAD and prefabrication concepts during the implementation of an everyday architectural design task, and also briefly compared AEC CAD with advanced CAD strategies which encapsulate computational design concepts. The outline methodology, *Mapping*

Integration, guided this 'reflection-in-action' and will be examined in more detail in the Discussion which follows.

DISCUSSION

Project work tested the hypothesis that computational design concepts are necessary to foster the integration of AEC CAD and prefabrication processes during the set up and design development stages of a three-storey residential building. The computational design concepts examined included the *design domain*, and *design rules* in the *declaratory format* and the *modular format*. A methodology, *Mapping Integration*, guided testing of the hypothesis during project work and forms the basis of a proposal for measuring integration intensity.

Design Domain - a template for reuse

The *design domain* is a useful computational design concept because it increases understanding and awareness of interactive relationships between elements, components and the envelope. The *design domain* is generally defined as a singular entity - a 'control rig' or 'jig' as perceived by Aish and Woodbury (2005, 11, 2007, 223) - but, as demonstrated here, it is better understood as a series of templates comprising the envelope, its components, and elements. The *design domain* describes set up of the templates which define these interactive relationships, thus allowing retrospective action to be taken to alter design intent. Another key advantage of the template is its tendency to reduce drafting by maximising the potential for re-use and sharing of the knowledge embedded in the templates themselves.

AEC CAD provided limited opportunities for setting up these frameworks, except at the component level, while advanced CAD provided every opportunity for setting up complex parametric associations between elements, components and the envelope but at

a high cognitive cost. Furthermore, though not utilised for this task, advanced CAD could have enhanced the dynamically interactive qualities of the *design domain* by encapsulating rules, checks, constraints and scripted 'knowledge patterns' within its *design domain* templates.

Thus, when setting up the *design domain* AEC CAD favoured ease of operability with reduced rigour, while advanced CAD traded-off ease of operability for rigour and precision. Clearly, for the *design domain* to become a concept which assists with integration of AEC CAD processes, ease of operability, rigour and precision should be possible at a reasonable cognitive cost.

Declarative Format – the challenges of scale

As a computational design concept, the *declarative format* improved intuitive understanding of interactive relationships between elements - relationships which are visually represented by nodes and wires - thus addressing Mitchell's (1989, 8) hope that the intelligibility of computational design processes could be improved. However, the flexibility of the *declarative format* is harder to verify. The project work demonstrated that nodes could fail, weakening operability. Other known problems include: 'scaling-up' which occurs when the graph becomes too large and complex, thus compromising operability and intelligibility (Burnett and al 1995, 46); and also 'separation' between the *visual programming graph's* interface and the geometric model which creates abstract separation between design intent and its implementation, as noted by Robert Aish (2005, 11). Hence, though useful to improve understanding of computational design concepts, these shortcomings of the *declarative format* impair its capacity to assist with integration of AEC CAD processes across the scales of element, component and envelope.

Modular Format – for programming and prefabrication

Mitchell's (1989, 8) additional assertion that the *modular format* could improve the intelligibility and flexibility of computational design processes was supported by its application to programming modules ensuring that they were self-contained, distinct with well-defined interfaces, reusable and modifiable. The project work demonstrated that such features could improve the prefabrication of *modular component assemblies*, an approach also promoted by Kieran and Timberlake:

> If we were to construct our buildings on site utilising preassembled components, the engineers could think in a more effective wholistic part-to-whole manner (Kieran and Timberlake 2004, 40).

Project work also demonstrated that advanced CAD encapsulates both the programming and prefabrication concepts of the *modular format*, a connection between concepts which suggests that further research would be productive. This suggest that the computational design concept of the *modular format* could successfully contribute to the integration of AEC CAD and prefabrication processes.

Mapping Integration

Following von Bertalanffy's and Simon's (2008, 1996) observations noted earlier, it is conjectured that an integrated computational design, AEC CAD and prefabrication process would possess the following features and attributes:

Openness

(O) *Openness*: interactions and exchange of feedback

(Io) *Interoperability:* exchange possible; error-free; sharing; re-use

(Co) *Collaboration:* participants work together to create or achieve objectives

Dynamic Interaction

(In) *Interactivity*:

i parametric associations

ii encapsulated knowledge

(Op) Operability:

i effective, user-friendly, reliable, supportable, maintainable

ii intelligible, low cognitive burden

iii affordance, '…the potential of the technology to enable the assertive will of its user' (Kalay 2004, 476)

Accordingly, a methodology is proposed, *Mapping Integration,* which would provide a graphical representation of the integration intensity of everyday design tasks (Figure 9).

Figure 9. Mapping Integration – sub-tasks' features assessed for integration intensity.

In the diagram above, integration intensity increases as values rise on the vertical scale. For instance, the interactivity (In_i, In_{ii}) levels of Task 1 (T_i) utilising the non-parametric methods of AEC CAD are very low, while the interoperability (Io_i, Io_{ii}) levels of Task 3 (T_{iii}) utilising advanced CAD are high. Sub-tasks representing the overlap of concepts, methods and tools are rated separately as shown. The goal of *Mapping Integration* is to further speculation and discussion about solutions. Following further research, the methodology will be developed and presented in more detail.

CONCLUSION

Utilising everyday architectural design tasks this research reflected on strategies to improve the integration of AEC CAD and prefabrication processes by implementing three computational design concepts: the *design domain*; the *declarative format*; and the *modular format*. These emergent computational design concepts have allowed insufficient time for practice to consider their utility and implications, thus it was appropriate to record findings through the methodology of 'reflection-in-action'.

The concept of the *design domain* enhanced intelligibility during set up of the model's templates because the act of intentionally associating multiple frameworks exposed interactive relationships between components and the whole. The added benefit of reduced drafting due to reuse of these templates supports the argument that this computational design concept should be more widely implemented in architectural practice.

Recommendations for improving the format of *design rules* in order to improve the flexibility and intelligibility of computational design processes, as advocated by Mitchell (1989, 8), highlighted the need for continuous review of these emergent tools. For instance, project work revealed that the *visual programming graph* which represents the *declarative format*, though accessible to computational design newcomers, is hindered by issues of scale, complexity and separation. Their recent introduction by AEC CAD suppliers, as an adjunct to the shortcomings of 'component based modellers' (Aish and Bredella 2017, 8), might encourage architects to be more watchful. Nevertheless, the *modular format* deserves further examination of properties which link programming efficiencies to prefabrication improvements, as demonstrated by the *modular component assembly* approach. Over the past 40 years advanced CAD and the manufacturing sector have developed methods and solutions to this approach deserving greater attention from the AEC sector.

In testing the hypothesis that computational design concepts can assist with integration of AEC CAD and prefabrication processes, this research has determined that more work needs to be done. In this vein, *Mapping Integration*, has been presented to foster the development of solutions appropriate to architects needs and the nature of their practice.

Computational design concepts integrated with AEC CAD and prefabrication practices

would encourage explicitly defined and intentionally devised workflows, thus requiring architects to interweave intuition with logic and rigour. Such a combination of concepts would prompt enhanced responses to the complex demands of the AEC sector and contribute to the emergence of an architecture fully engaged with the processes of design and production.

REFERENCES

Aconex. 2017. "What is BIM." Aconex, accessed 16 November. https://www.aconex.com/what-is-BIM.

Aish, Robert. 2005. "From Intuition to Precision." eCAADe 23, Lisbon.

Aish, Robert, and Nathalie Bredella. 2017. "The evolution of architectural computing: from Building Information Modelling to Design Computation." *Architectural Research Quarterly* 21 (1).

Burnett, M, and et al. 1995. "Scaling up Visual Programming Languages." *Computer* 28 (3):45-54.

Burry, Jane. 2007. "Mindful spaces: computational geometry and the conceptual spaces in which designers operate." *International Journal of Architectural Computing* 5 (4):611-624.

Burry, Mark. 2011. *Scripting Cultures: Architectural Design and Programming*, *AD Primers*. Hoboken, NJ: John Wiley & Sons Ltd.

Carpo, Mario. 2011. *The Alphabet and the Algorithm*. Cambridge, MA: MIT Press.

Davies, Colin. 2005. *The Prefabricated Home*. London: Reaktion Books.

Davis, Daniel, Jane Burry, and Mark Burry. 2011. "Untangling Parametric Schemata: Enhancing Collaboration through Modular Programming." Designing Together - CAADFutures 2011, Liège.

Eastman, Charles, David Fisher, Gilles Lafue, Joseph Lividini, Douglas Stoker, and Christos Yessios. 1974. An Outline of the Building Description System. Pittsburgh, PA: Carnegie Mellon University, Institute of Physical Planning.

Habraken, Nicholas J. 1999. *Supports: an alternative to mass housing*. 2nd ed. Gateshead, Tyne & Wear: Urban International Press. Original edition, 1961.

Hasselbring, Wilhem. 2002. "Component-Based Software Engineering." In *Handbook of Software Engineering and Knowledge Engineering: Emerging Technologies*, edited by Shi Kuo Chang, 289-305. Singapore: World Scientific.

Kalay, Yehuda E. 2004. *Architecture's New Media: principles, theories and methods of Computer-Aided Design*. Cambridge, MA: MIT Press.

Kieran, Stephen, and James Timberlake. 2004. *Refabricating Architecture: How Manufacturing Methodologies are Poised to Transform Building Construction*. New York, NY: McGraw Hill Professional.

Mitchell, William J. 1989. "A New Agenda for Computer-Aided Design." Electronic Design Studio, Boston, MA.

Park, Jin-Ho. 2005. "Demountable and interchangable construction system: R. M. Schindler's Panel Post Construction." The 2005 World Sustainable Building Conference, Tokyo.

Russell, Barry. 1981. *Building Systems, Industrialisation and Architecture*. London: John Wiley & Sons.

Schodek, D, M Bechthold, K Griggs, K M Kao, and M Steinberg. 2005. *Digital Design and Manufacturing: CAD/CAM Applications in Architecture and Design*. Hoboken, NJ: John Wiley & Sons Inc.

Schön, Donald A. 2008. Part 1: Professional Knowledge and Reflection-in-Action. In *The Reflective Practitioner: How Professionals Think In Action*. New York, NY: Basic Books Original edition, 1983.

Simon, Herbert A. 1996. *The Sciences of the Artificial*. 3rd ed. Cambridge, MA: MIT Press. Original edition, 1969.

Smith, R E. 2010. *Prefab Architecture: A Guide to Modular Design and Construction*. Hoboken, NJ: John Wiley & Sons.

Terzidis, Kostas. 2006. *Algorithmic Architecture*. Oxford: Elsevier.

von Bertalanffy, L. 2008. "An Outline of General System Theory." In *Emergence, Complexity and Self-Organisation: Precursors and Prototypes*. Marblehead, MA: ISCE Publishing (accessed 12 September 2017).

Woodbury, Robert, Robert Aish, and Axel Kilian. 2007. "Some Patterns for Parametric Modelling." ACADIA 2007, Halifax, Nova Scotia.

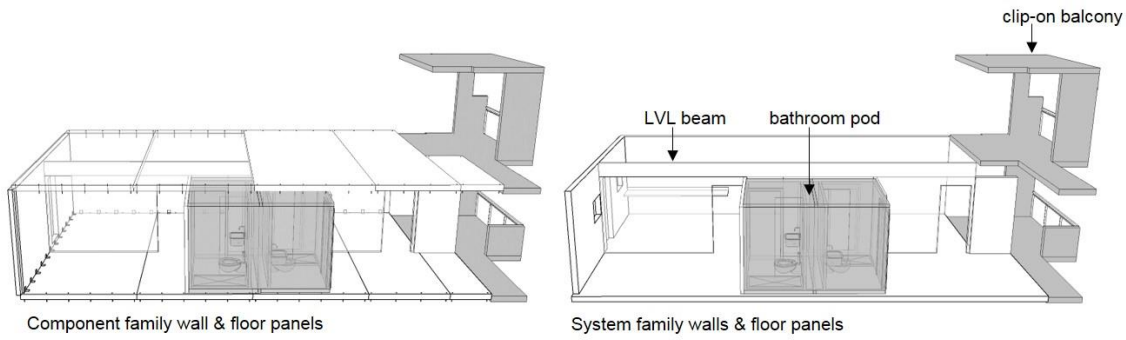

Figure 1. Typical set-up AEC CAD: 'grouped' modules - unit, bathroom pods, clip-on balconies (Autodesk Revit).

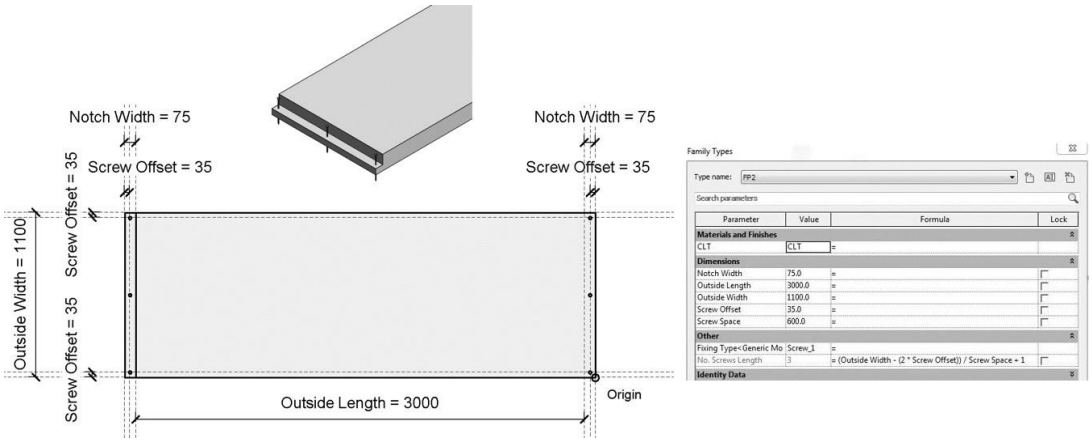

Figure 2. Floor panel design domain - a template for floor panel types (Autodesk Revit).

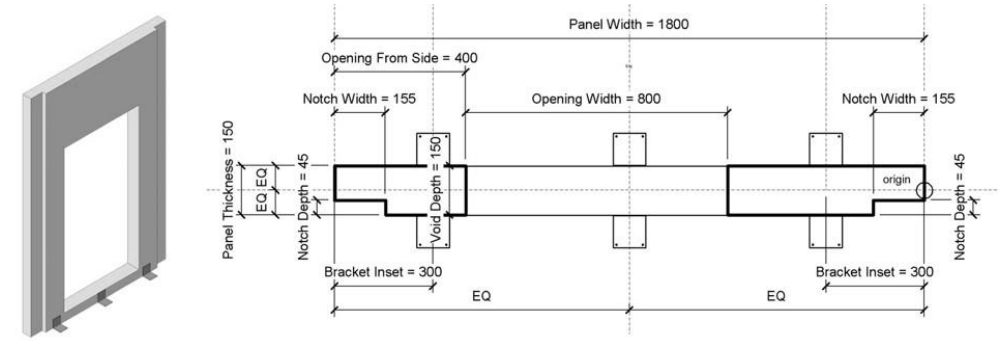

Figure 3. Wall panel component design domain: reference planes, associated dimensions, and geometry (Autodesk Revit).

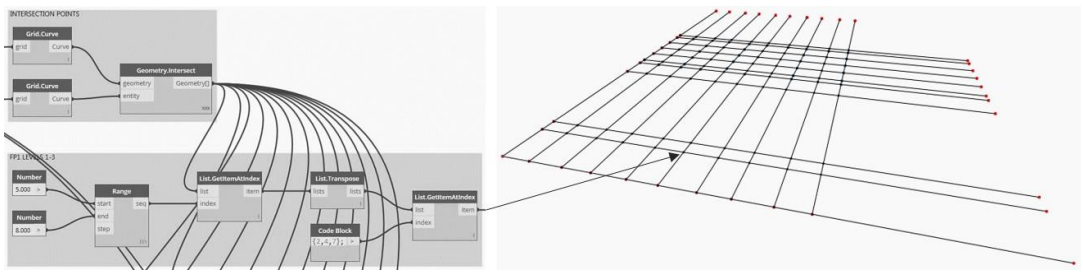

Figure 4. Set-up of the model's design domain to find grid intersection coordinates (Autodesk Dynamo).

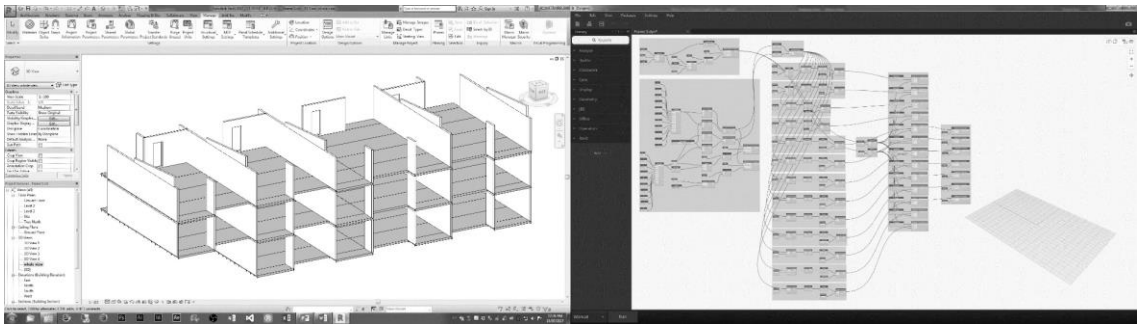

Figure 5. Automatic placement and rotation of wall and floor panels based on coordinates projected to locations at each level (Autodesk Revit & Dynamo).

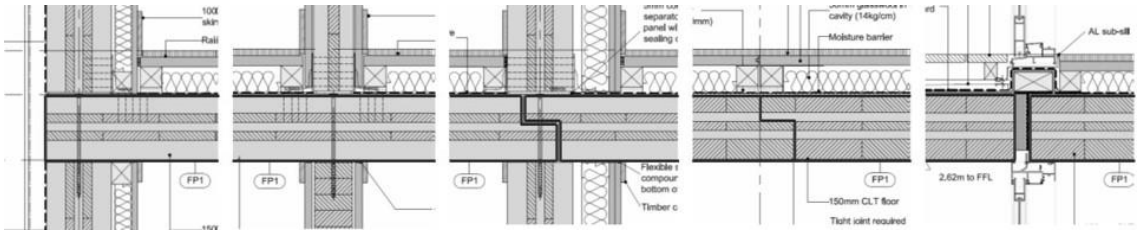

Figure 6. Floor & wall panel interfaces (background details by Kate Humphries, UQ).

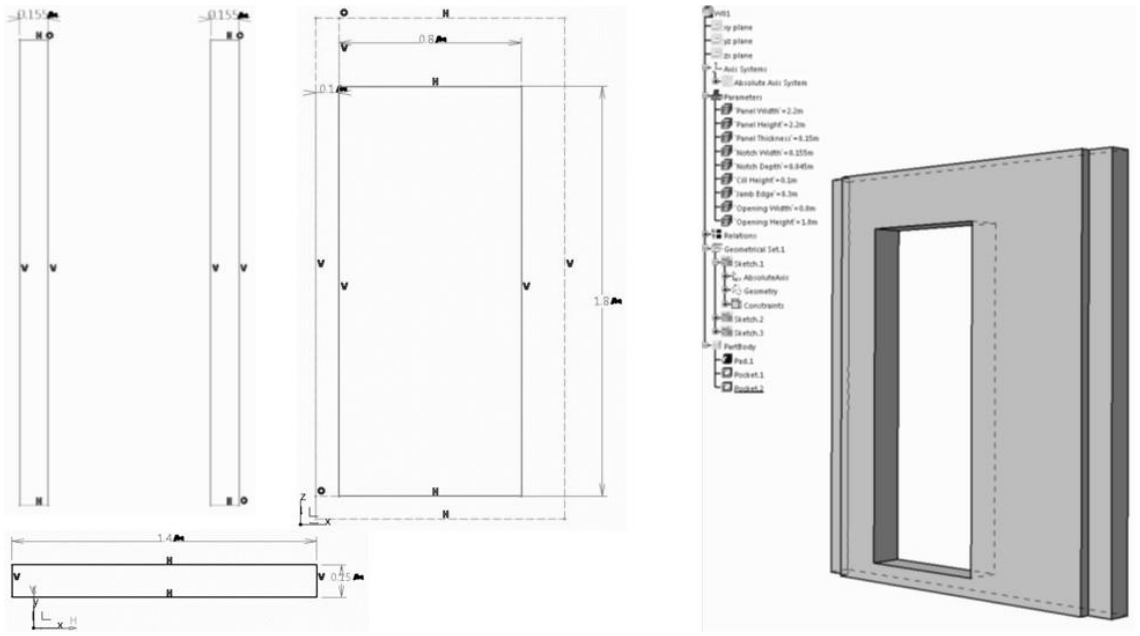

Figure 7. Framework sketches, 'feature based' product assembly & hierarchical tree (Digital Project/CATIA).

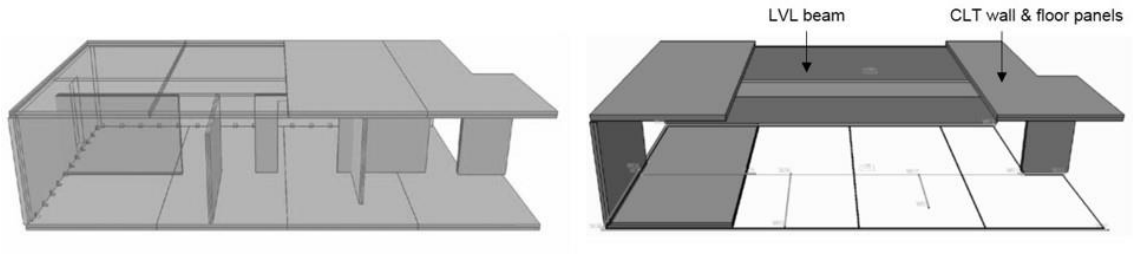

Figure 8. Grids for each CLT wall and floor panel component, Glulam beam, and automated wall bracket placement to unit type (Digital Project/CATIA).

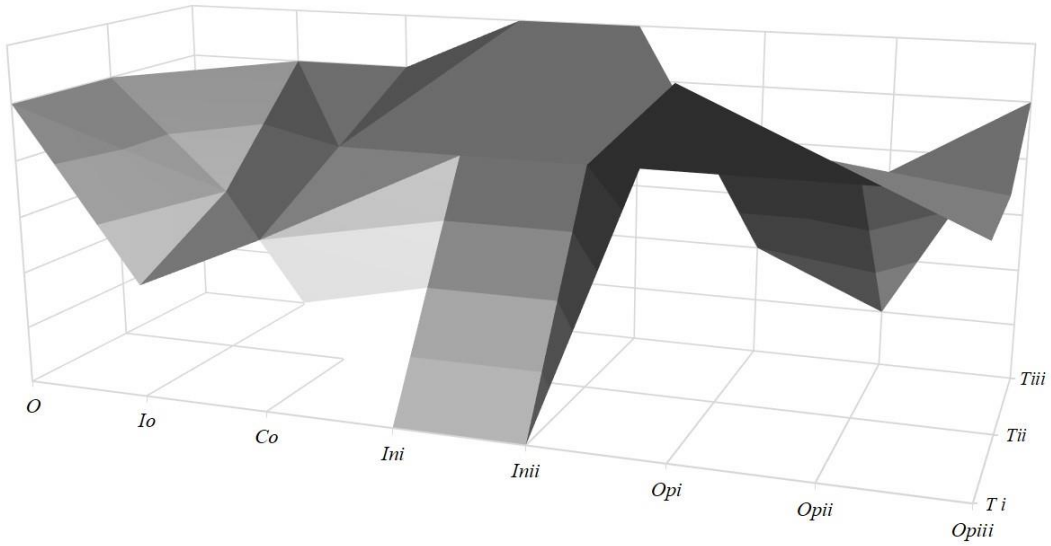

Figure 9. Mapping Integration – sub-tasks' features assessed for integration intensity.